Week	Date	Teaching	Attended
5	**Feb 2013**	Lab 4: Vulnerability Analysis & Enumeration/Toolkit 4	

Aim: The aim of this lab is to investigate Network Scanning, System Enumeration, and Vulnerability Analysis, using tools such as Nmap, Nessus and Tripwire. The lab has been built around using two virtual machines. UBUNTU and a WINDOWS2003 VMs. These can be either hosted on a local PC or run on the Napier virtual environment at vc2003.napier.ac.uk.

Time to complete:
4-5 hours (Two supervised hours in the lab, and two to three hours, unsupervised).

Activities:

- **Complete Lab 4**: Vulnerability Analysis & Enumeration
  .pdf from WebCT or http://www.dcs.napier.ac.uk/~cs342/CSN10102/Lab4.pdf

- Complete the **End Of Unit Tutorial Questions** for this unit, with the **NetworkSims** Test Engine, and online questions at:
  http://asecuritysite.com/security/tests/tests?sortBy=sfc10

Learning activities:
At the end of these activities, you should understand:

- How to use Nessus to automatically scan for vulnerabilities.
- The security issues with services such as Telnet and FTP, and how to mittigate against them.
- How to use Host IDS such as Tripwire to monitor for intrusions.
- How to integrate TCPDump into the toolkit.

Reflective statements (end-of-exercise):
Why are protocols such as Telnet, FTP, and HTTP vulnerable to attacks?
Why are these protocols not more secure?
In an organisation, why would vulnerability analysis tools, such as Nessus, be used to audit systems on a regular basis?

Source code used:
http://buchananweb.co.uk/toolkit.zip

Lab 4: Vulnerability Analysis and Enumeration

Rich Macfarlane 2013

4.1 Details

Aim: The aim of this lab is to investigate protocol vulnerabilities, through the use of nmap, Nessus and Tripwire.

NOTE: DO NOT use techniques from this lab on any external networks. This lab is to be performed in a sandboxed environment, using a private address space only.

4.2 Activities

> To assist with this part of the lab and the following demo can be used:
> http://bit.ly/eZPDad

Run the Windows Server 2003 virtual image **WINDOWS2003**. Log into the Windows server as User name: **Administrator**, Password: **napier**).

Within the VM WINDOWS2003, run a **Command Prompt Window** (Start>Run>cmd) and determine the servers IP Address using **`ipconfig`**.

Run the Linux virtual image **UBUNTU**. Login as User: **napier**, Password: **napier123**).

Within the virtual image UBUNTU, open a **Terminal Window** (Applications>Accessories>Terminal) and determine the servers IP Address using **`ifconfig`**.

> Complete the IP Addressing diagram in Figure 1 or Figure 2 below, depending on which architecture you choose to use.

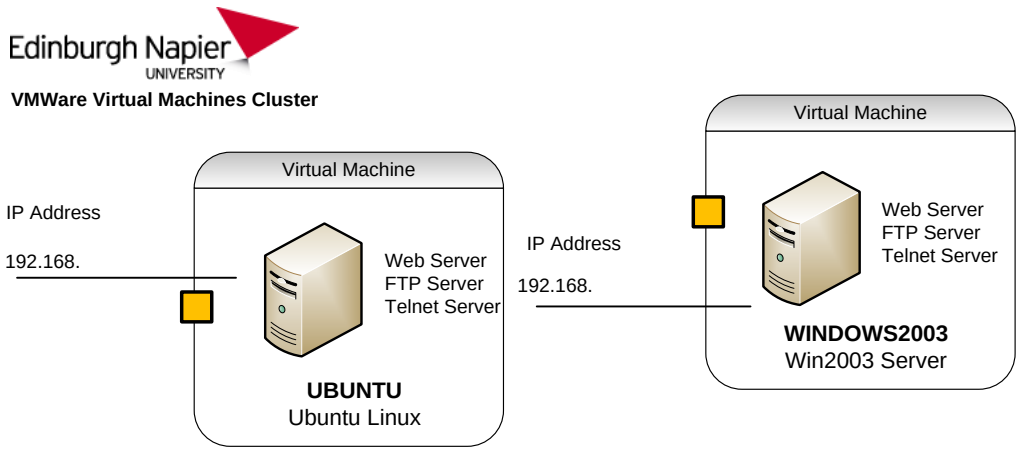


Figure 1 – Cloud-based Lab IP Addressing

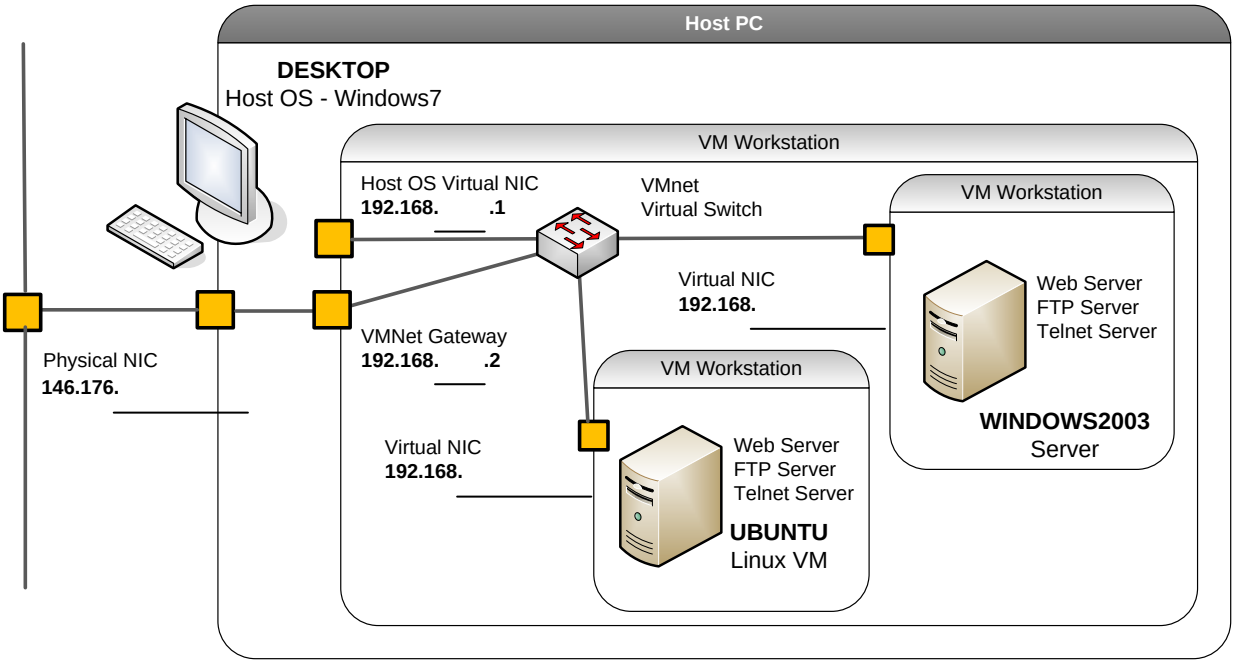
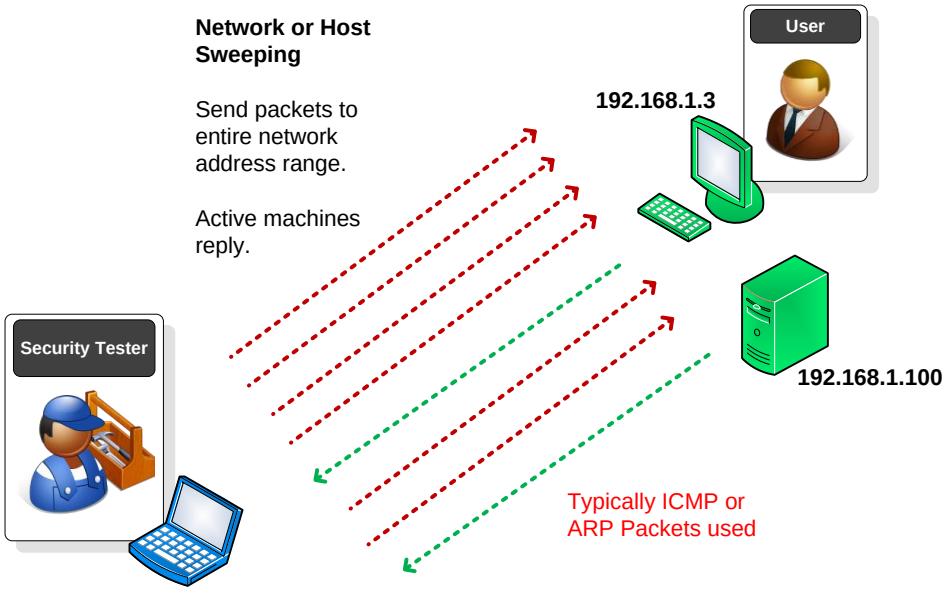


Figure 2 – Local Lab Architecture and IP Addressing

Network Scanning - Host Discovery

The first step in most Security Assessments is to find machines on the target network. At this stage the identity of the systems (routers, PCs, Servers) or what services they are running are not being sought, but only how many target systems are live, and what are their IP Addresses. These will then be enumerated later.

A **Host Sweep** is a simple method of finding target systems. Packets such as ICMP echo or ARP requests are sent across the network, to which any hosts at the addresses reply with ICMP echo-reply or ARP reply packets, as illustrated below.



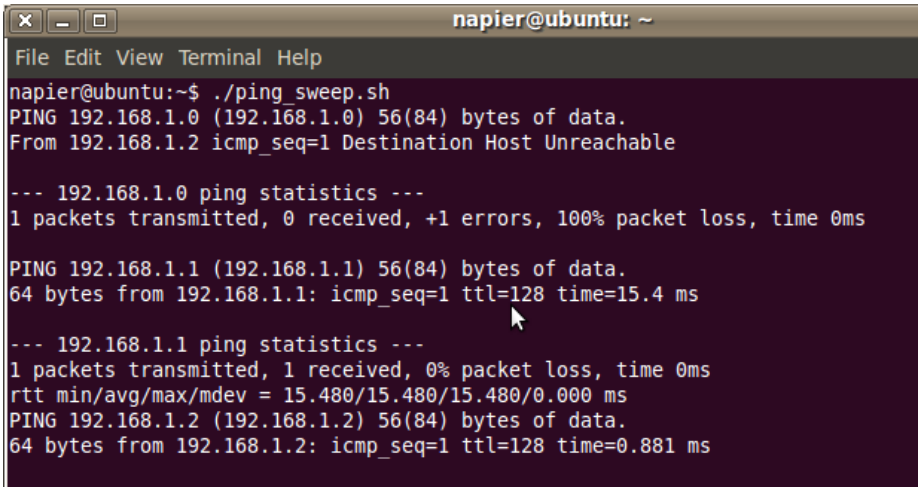
Ping

The **`ping`** tool is the most well know for checking connectivity to other systems. It uses ICMP packets.

You could manually ping each address on the target LAN to find target machines, but it is easy to automate tools using scripting languages. To automate the use of ping to find targets, we can use a simple **bash shell script** on the command line to send 1 ICMP request packets to the first 10 IP Addresses on the LAN, such as:

```
for i in {0..10}; do ping -c 2 192.168.100.$i; done;
```

The results should look like the following:



This is fairly messy. We can clean up the output, by only reporting the IP addresses of the machines that reply, using grep and awk in a script such as the following:

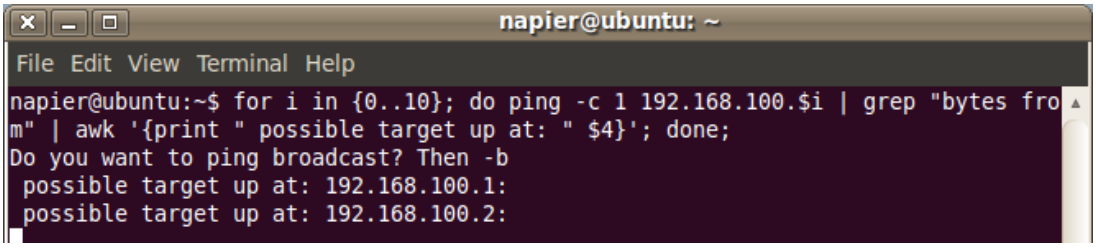
```
for i in {0..10}; do ping -c 2 192.168.100.$i | grep "bytes from"
| awk '{print " possible target up at: " $4}'; done;
```

Run on a range of IP Addresses in your subnet, and report the findings below:

☞ How many hosts are alive on the subnet? List some below:

☞ What might prevent the ICMP sweep from reporting hosts?

The output should look similar to the following:

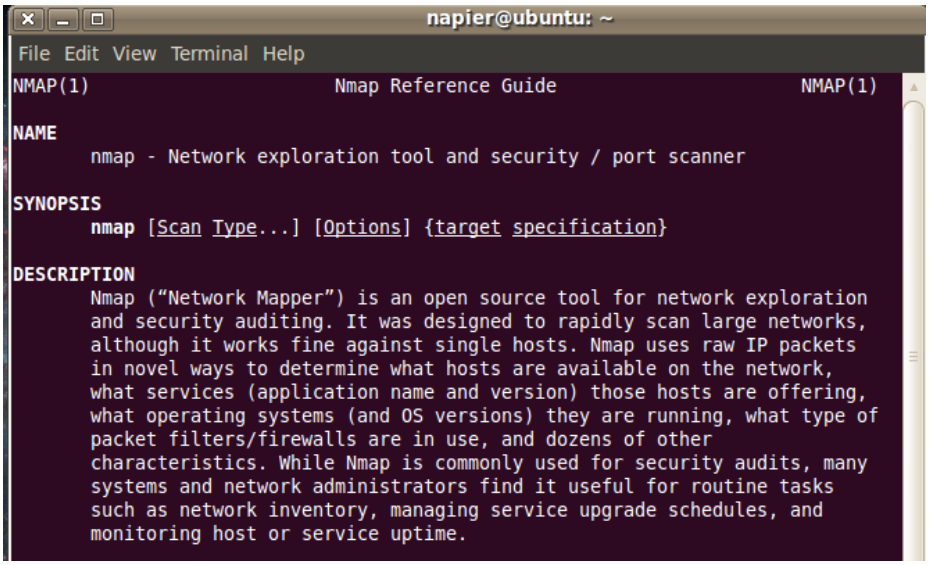


Nmap

Nmap is one of the most popular network scanning tools. It is widely available, for Windows and Linux/Unix platforms, and has both a Command Line Interface (CLI) and a Graphical User Interface (GUI). We will use the CLI in this lab.

The **nmap** users manual is available at:
http://nmap.org/book/man.html

Or from UBUNTU try **`man nmap`** at the command line, to get the manual page as shown below. (man can be used for almost all commands/tools at a Linux command line) Use <CTRL+F> or <SPACE> gives the next page, <CTRL+B> the previous, and <RETURN> the next line. Use **:q** to quit out of man.

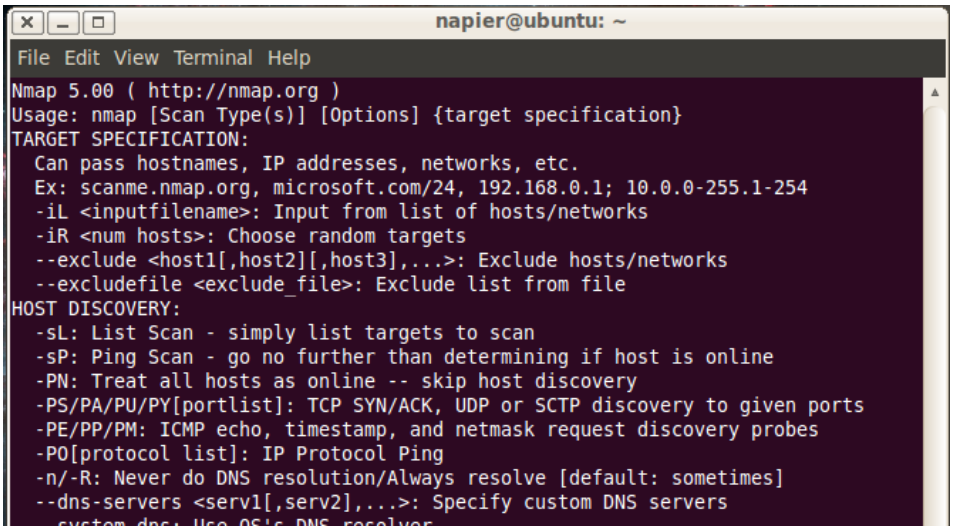


From UBUNTU, check some of the many parameters/flags used with the **nmap** network scanner:

```
sudo nmap -? | more
```

Note: The output from **`nmap`** is being piped into the **`more`** command, as shown below.

Within more <CTRL+F> or <SPACE> gives the next page, and <RETURN> the next line. (More details on `more` can be found using **`man more`**)



From UBUNTU, use the **nmap** network scanner to find any host systems which are active on the same subnet as your UBUNTU machine:

```
sudo nmap –sP –r 192.168.x.0/24
```

Note: The **`–sP`** specifies a Host Sweep, **`–r`** specifies they are listed in order.
The **`/24`** specifies the CIDR notation for a class C subnet (256 hosts). If you use `/1` nmap will try and scan over 2 billion host addresses... so please don't!

- How many hosts are alive on the subnet? List some below:

- What Information is being reported by the Host sweep?

- Has your WINDOWS2003 server been discovered?
  YES/NO
- Can you tell the OS of the hosts?
  YES/NO
- Would this discover host systems which are shut down?
  YES/NO

Perform the nmap scan again, while using Wireshark to capture the packets nmap is using to perform the host sweep. (**`sudo wireshark &`** from the command line)

- Which protocol is nmap using to discover hosts?

Enumeration - Operating System Fingerprinting

Enumeration is the gathering of information about target hosts. After discovering live target systems, we want to identify which machines are running which OS's, as illustrated below.

A useful feature of **nmap**, is determining the operating system of hosts on the network. It performs active OS fingerprinting by sending packets to the target system.

- Which nmap flag can be used to enable OS Detection?

Nmap sends a combination of packets to various ports on the target system, and the response packets of the OS are like a fingerprint. Depending on the ports open on the host, the results can vary from very accurate, to a broad range of possible OSs.

Perform an OS Fingerprint Scan on some of the hosts discovered on the network, using a command such as:

```
sudo nmap -O hostIpAddress
```

- Does nmap return an exact match for the OS fingerprint?

  YES/NO

- If not, look up the nmap host discovery flags, and find a flag to make nmap return a best guess. What is the flag used and what did it guess the OS to be?

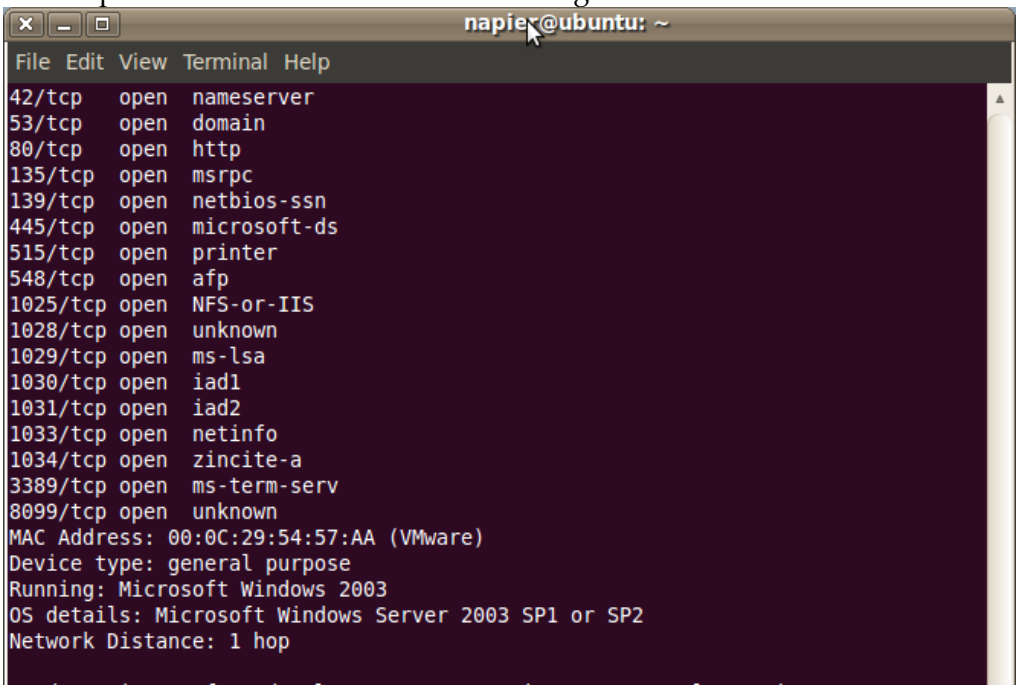
Perform an OS Fingerprint Scan of your WINDOWS2003 VM.

- Does nmap return an exact match for the OS fingerprint?

  YES/NO

- What does it return for the OS?

The output should be similar to the following:



Enumeration – Application Fingerprinting

Application Fingerprinting or **Banner Grabbing** covers techniques to enumerate OSs and Applications running on target hosts. An attacker or security tester would be specifically looking for versions of Applications or OSs which have vulnerabilities.

Nmap can be used to check applications and versions for network services running on the target for the open ports it finds during a port scan. The **`-sV`** flag can be used to do application and version detection.

```
sudo nmap -sV hostIpAddress
```

Perform an Application Fingerprint Scan of a Linux system you discovered (you can scan your own UBUNTU system if running local VMs)

- Does nmap return applications and versions of the Network Services running on the target?

  YES/NO
- Which web server is running and which version?

Perform an Application Fingerprint Scan of your WINDOWS2003 system.

- Does nmap return applications and versions of the Network Services running on the target?

  YES/NO
- Which web server is running and which version?

Telnet is another tool commonly used for banner grabbing. Once open ports have been found using a scanner, Telnet can be used to connect to a service and return its banner.

From UBUNTU connect to the Web Server (port 80) on WINDOWS2003 using Telnet:

```
telnet w.x.y.z 80
```

and then send the HTTP `OPTIONS` command to the web server:

```
OPTIONS / HTTP/1.0
```

- What is returned and how can this be used to fingerprint the WebServer?
- Which WebServer is running and which version?

Similarly, other **HTTP** commands such as `HEAD` (get a HTML page header) and `GET` (get the whole HTML page) can be used to footprint a web server.

```
HEAD / HTTP/1.0
GET / HTTP/1.0
```

Now try banner grabbing the web server on UBUNTU from WINDOWS2003, using Telnet in the same way.

☞ Which WebServer is running on UBUNTU, and which version?

Enumeration – Windows WMIC

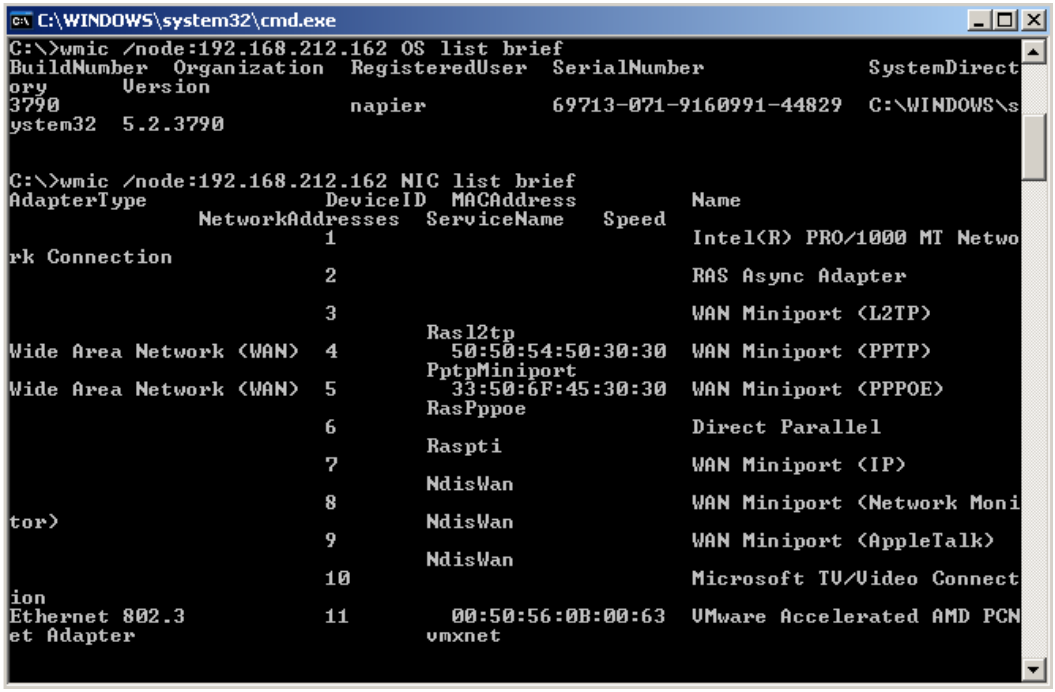
To assist with this part of the lab and the following demo can be used:
http://bit.ly/fyyF0B

Windows Management Instrumentation Command-line (WMIC), allows the use of Windows Management Instrumentation (WMI) from the Windows command line. WMI is a model for accessing management information, which can be used by applications.

From WINDOWS2003, ask your neighbor for their WINDOWS2003 host IP Address, or use a discovered Windows host on the network (if not using cluster enumerate your own server). Enumerate the target hosts details with the following:

```
wmic.exe /node:w.x.y.z CPU list brief
wmic.exe /node:w.x.y.z NIC list brief
wmic.exe /node:w.x.y.z OS list brief
wmic.exe /node:w.x.y.z SHARE list brief
```

The output should be similar to the following:



☞ What is the MAC address of the windows host?

☞ Which Shares are found on the host?

- ☞ Outline some other details:
- ☞ What other options are available in WMIC?

Get your neighbour to check his MAC address using **`ipconfig/all`**

- ☞ Did you enumerate his MAC address successfully?

YES/NO

Ask your neighbour to change their host name, and rescan it and determine the new name.

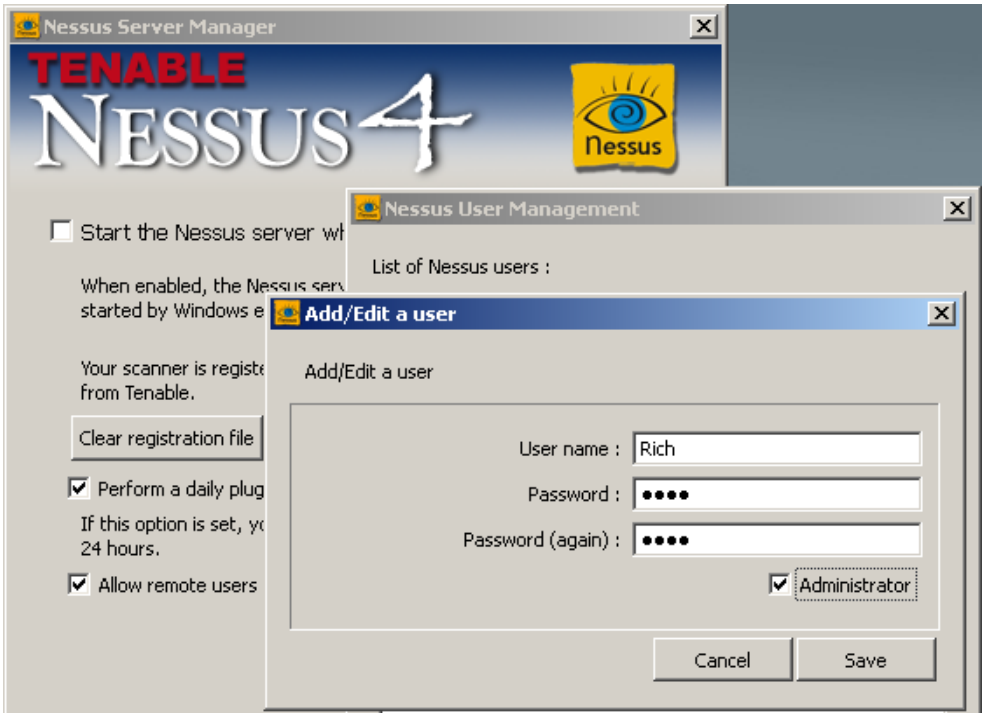
- ☞ Did you enumerate his neighbours new hostname?

YES/NO

Vulnerability Scanning

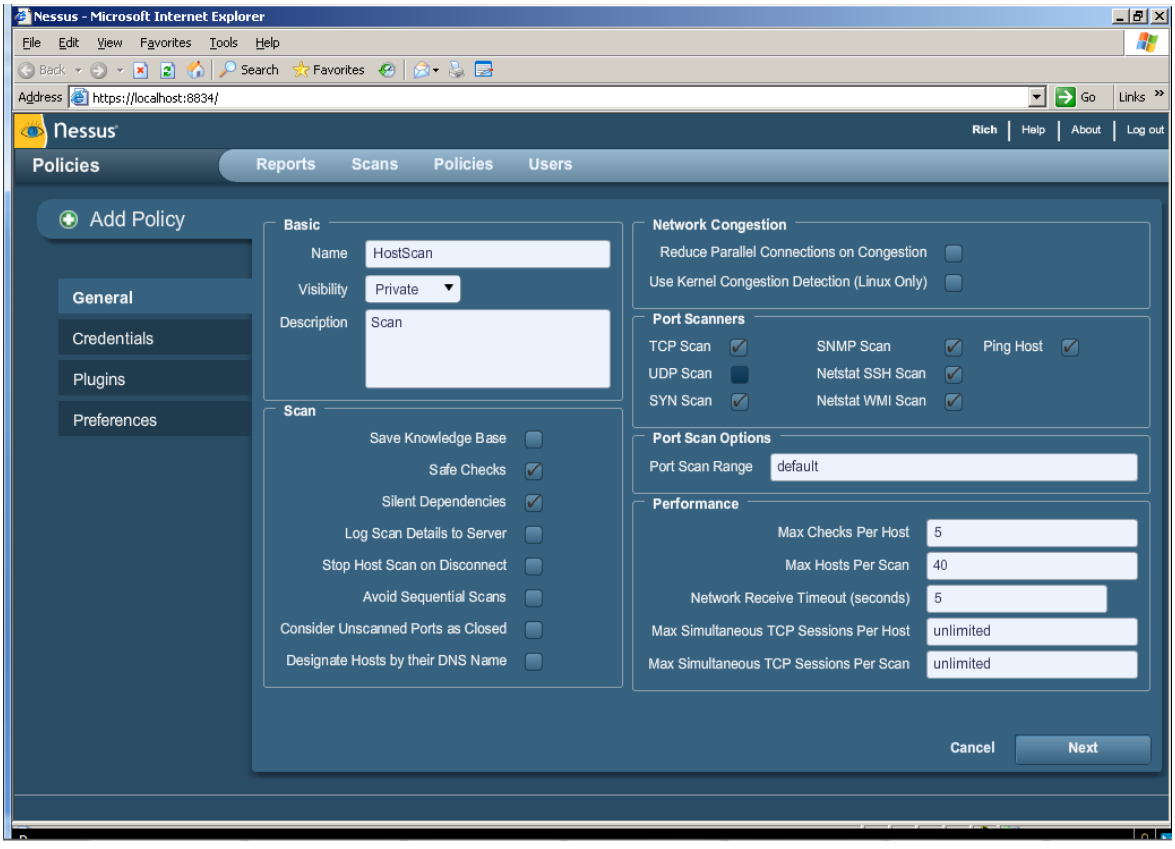
Vulnerability Scanners searches for known vulnerabilities in services it find on host systems. **Nessus** is one of the best known vulnerability scanners, and is available for most platforms including Windows, Linux, Unix, and Mac.

From WINDOWS2003, run the **Nessus Server Manager** and create a new user account, as shown below. Click the **Start Nessus Server** button if the server is not running.



On WINDOWS2003, run the **Nessus web client**, and log in with the user you created, as shown below. (you may have to install the latest version of Flash)

Create a new **Policy** called **HostScan**. Select all the Port Scanning items, such as for TCP Scan, Syn Scan, SNMP Scan, and the other defaults, as shown below.



Click **Next** through the options pages, and click **Finish** to save the Policy.

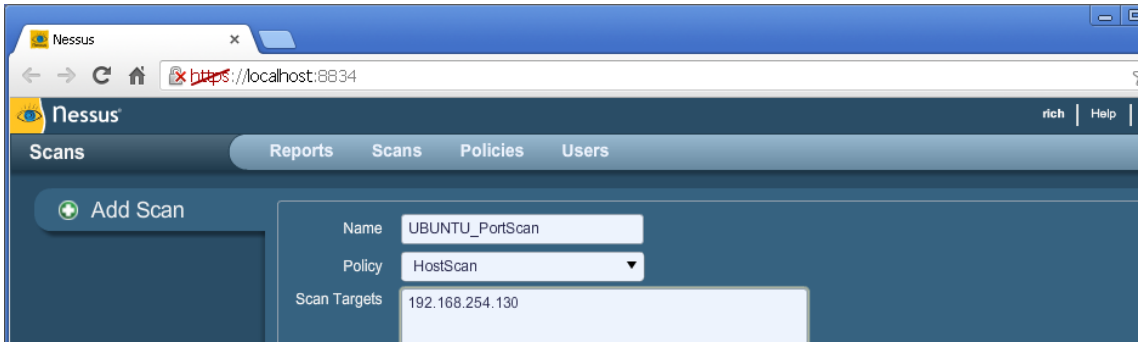
On UBUNTU, monitor the scan using TCPDump:

```
sudo tcpdump -i eth4
```

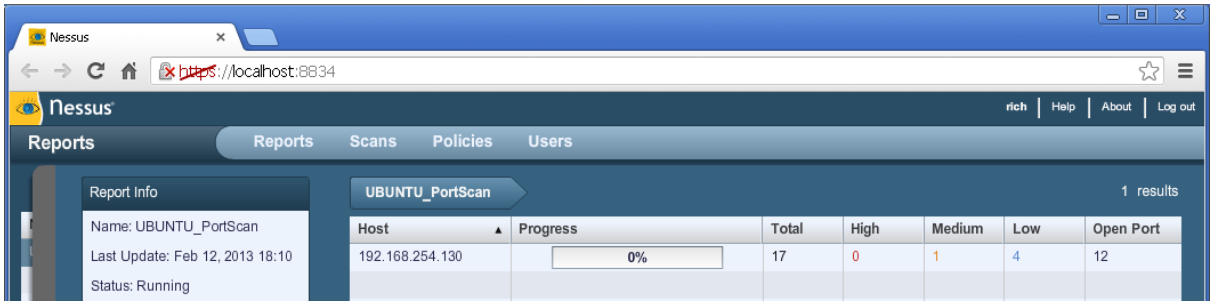
or using Wireshark:

```
sudo wireshark -i eth4 &
```

L4.1 On WINDOWS2003, Create a new **Scan** using the HostScan **Policy** you created, and then add the UBUNTUs IP Address as a target system.

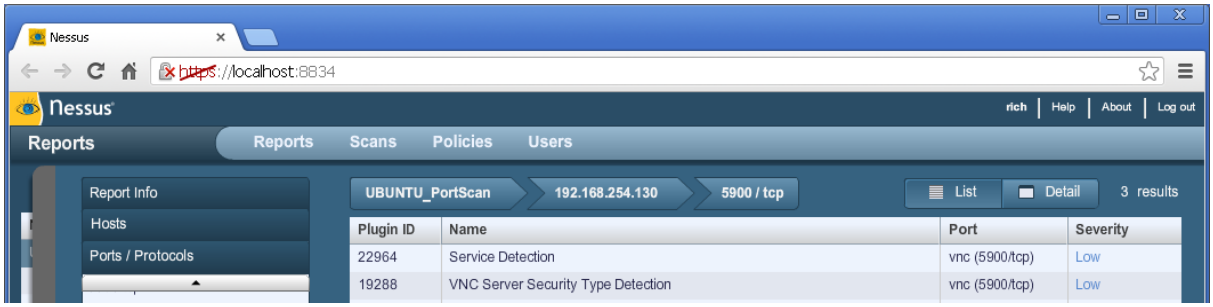


Click **Launch Scan** to run the new scan against the UBUNTU host. Nessus will scan the target host for services which are running, enumerate the applications and versions, and report on any vulnerabilities.
Double click the scan, and the progress can be viewed, as shown below.



Nessus produces a report on the running services, and any associated vulnerabilities it has found.

Double click on a service to see detail on the possible risks Nessus has found:



Double click on a vulnerability to see detail and possible security solutions:

☞ From the Nessus Report for the UBUNTU scan, what are the Services found? (including any type and version information)

Telnet Weaknesses

One of the services which is reported on, and is weak is **Telnet**. Run Wireshark as root on UBUNTU, and start a capture on the eth interface.

From WINDOWS2003 Telnet to UBUNTU from the command line.

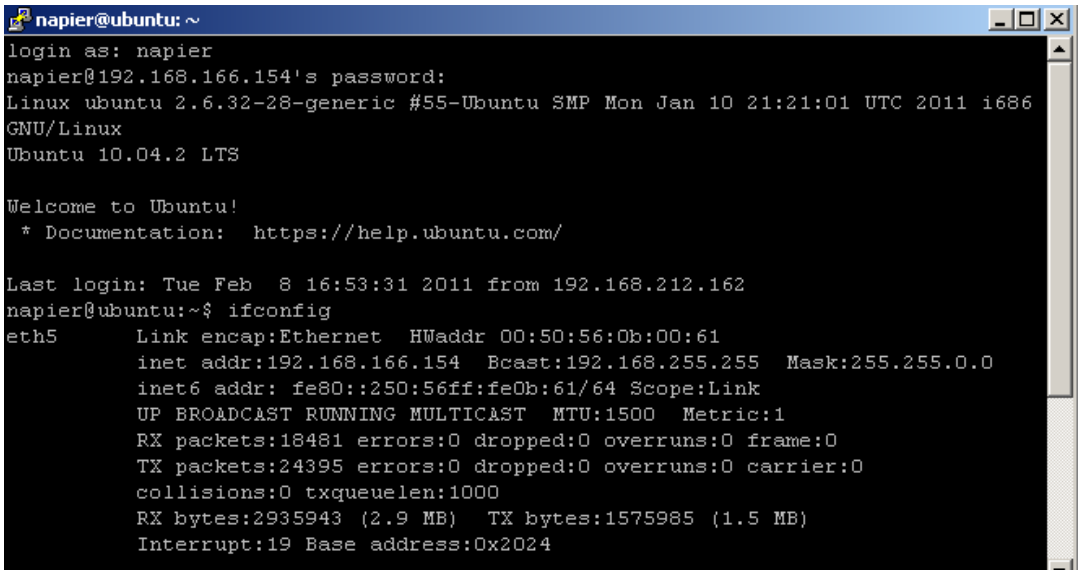
Stop the Wireshark capture and observe the Telnet login packets in Wireshark, and determine the user name and password, by reassembling the Telnet Traffic using the **Follow TCP Stream** function.

SSH

Next install the Secure Shell (SSH) server on UBUNTU with:

```
sudo apt-get install openssh-server
```

Now from WINDOW2003, login with an SSH client (download Putty if necessary), while capturing the login process in Wireshark on UBUNTU. The SSH login is shown below.



On UBUNTU, stopFrom the wireshark capture and reassemble the session.

☞ What can be observed from login with SSH?

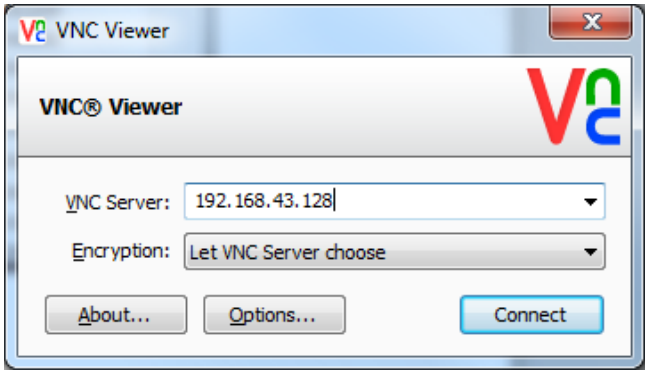
VNC

From WINDOWS2003, run a Nessus scan with the **Misc** plug-in, and show that it produces a**VNC Server Unauthenticated Access** vulnerability.

On WINDOWS2003, download a VNC client, from the following:
`http://www.realvnc.com/cgi-bin/download.cgi`

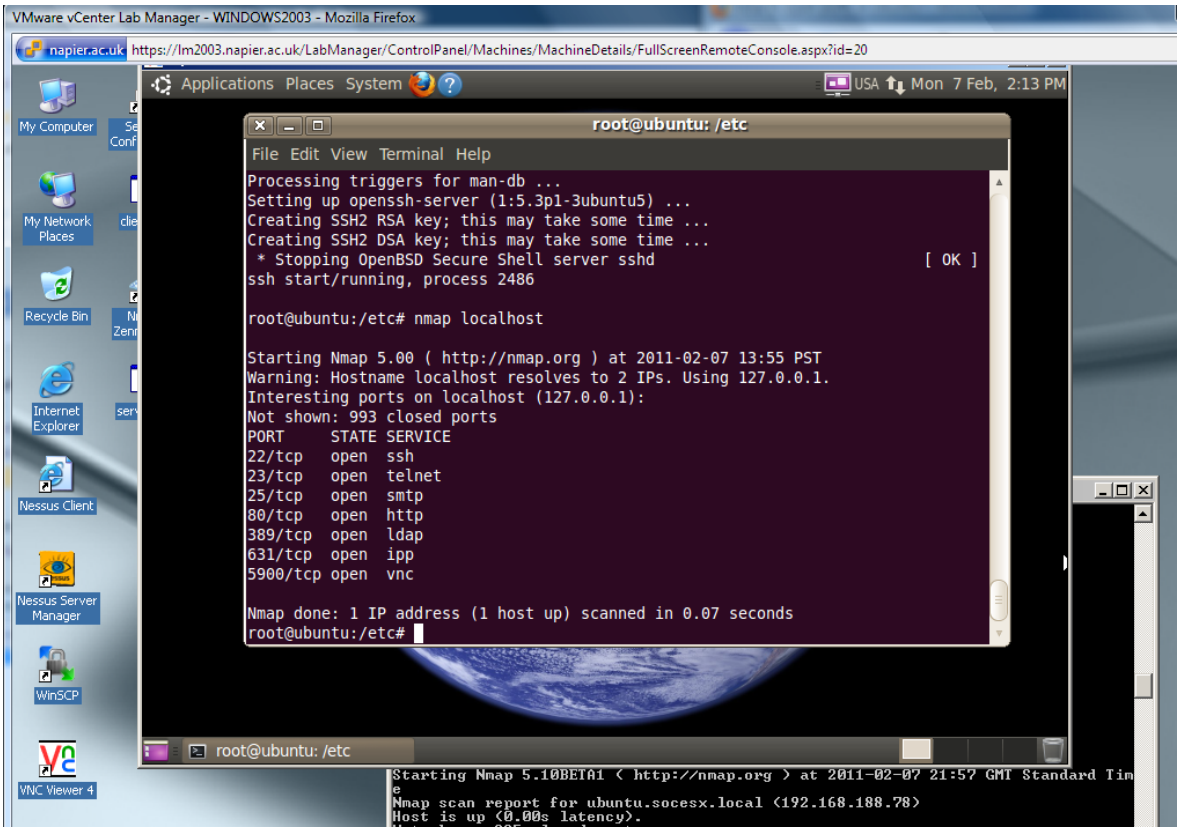
From WINDOWS2003, connect to UBUNTU using the VNC client, as shown below:

Note: A dialog may appear on the server, which requires you to allow access to the VNC Client program.

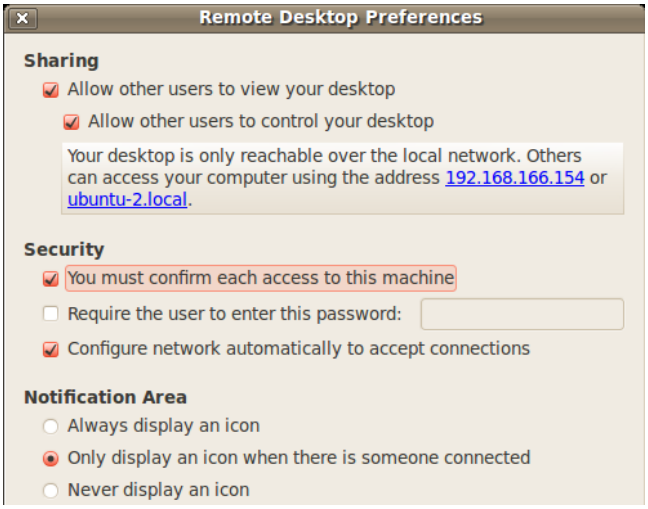


VNC Viewer Client Application

Now prove that there is non-authenticated access to UBUNTU as shown below:



On UBUNTU, select **System>Preferences>Remote Desktop**. Check the **You must confirm each access to this machine** checkbox, as shown below.



☞ How has the VNC access been changed?

FTP

On WIN2003 create a new Nessus Policy called **FTPScan**, and delete the Port Scanning options, and select Next. After this select the **FTP** plug-in as shown below.

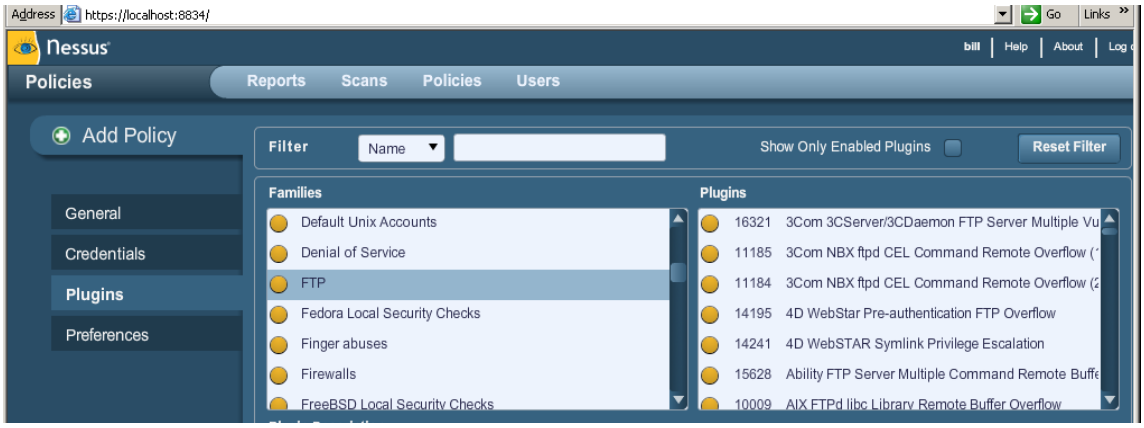


Figure: FTP scan with plug-in

Run the new scan against UBUNTU.

☞ What is the FTP vulnerability reported?

Secure FTP

Next, from UBUNTU, we will run a secure FTP server (SFTP). To do this install the following:

```
sudo apt-get install rsplib-tools
sudo apt-get install vsftpd
```

and edit the VSFTPD config file with:

```
sudo nano /etc/vsftpd.conf
```

and add the following lines at the end of the file (where we have default digital certificate files and a private key):

anonymous_enable=YES
local_enable=YES
rsa_cert_file=/etc/ssl/certs/ssl-cert-snakeoil.pem
rsa_private_key_file=/etc/ssl/private/ssl-cert-snakeoil.key
ssl_enable=YES
force_local_logins_ssl=YES
force_local_data_ssl=YES
ssl_tlsv1=YES
ssl_sslv2=YES
ssl_sslv3=YES

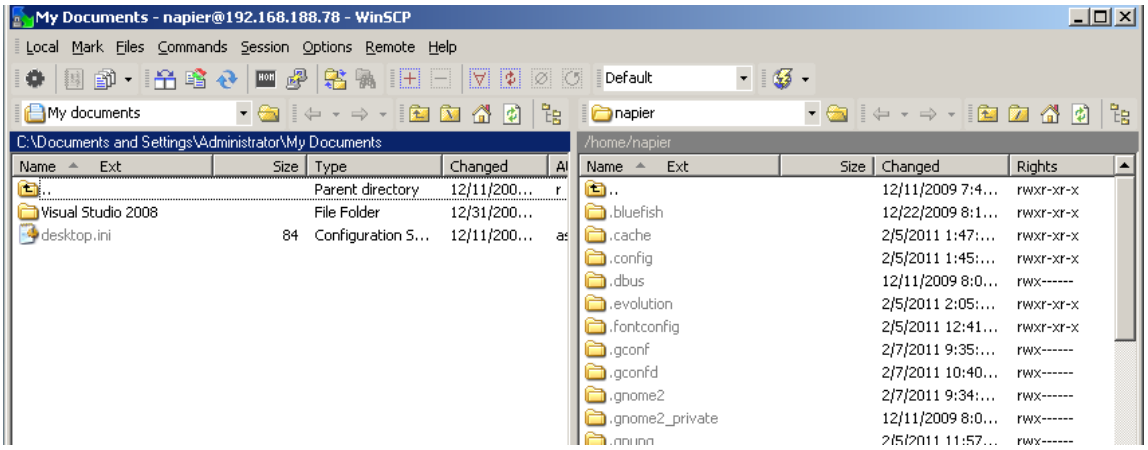
Finally start the FTP daemon:

```
server vstpftpd start
```

Next, from WINDOWS2003, install a SFTP Client, such as WINSCP.

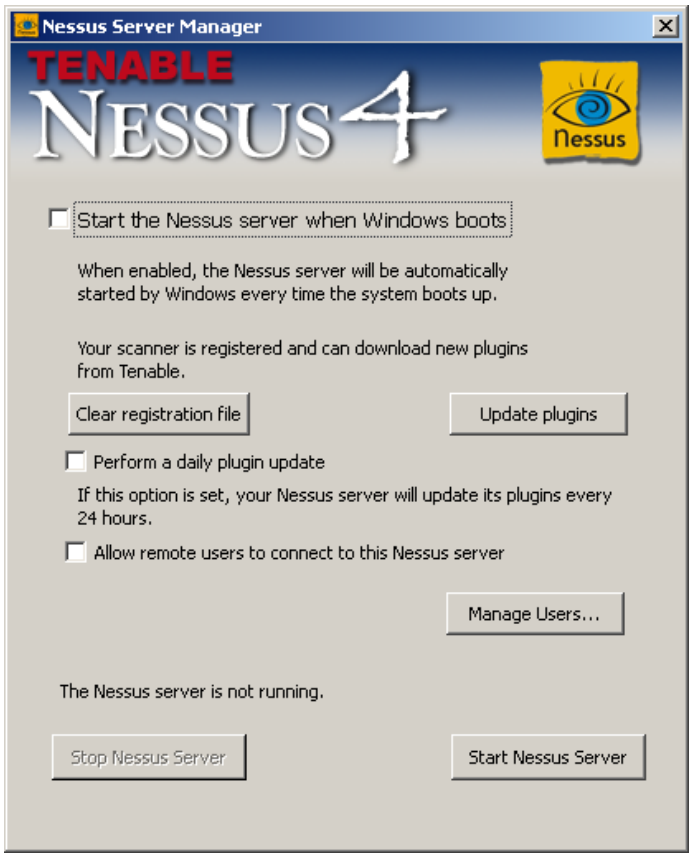
WinSCP can be downloaded from:
http://winscp.net/eng/index.php

From WIN2003, connect to the SFTP server on UBUNTU using the SFTP client (Figure 4.7). Now transfer a file from WINDOWS2003, and provide that it exists on UBUNTU (/home/napier).



Secure FTP access to UBUNTU

From WIN2003, run the Nessus Server Manager, and **Stop the Nessus Server.**



4.3 Tripwire Tutorial

Tripwire is a Host IDS which can monitor a host system for signs of intrusion. It is classified as a File Integrity Monitor, and works by comparing the current state of a system to a baseline or snapshot of the system.

	On-line video demo of the tripwire lab: http://buchananweb.co.uk/adv_security_and_network_forensics/tripwire/tripwire.htm

L4.2 Run UBUNTU (User name: napier, Password: napier123). Within the virtual image, run a Terminal and determine its IP address using **ifconfig**.

L4.3 Create Tripwire Filesystem Snapshot

Go to the **/etc/tripwire** folder, and view the **twpol.txt** file.

Next run the following commands to create the tripwire policy file and the tripwire database (use **napier123** as the site and local encryption keys):

```
sudo twadmin --create-polfile --cfgfile ./tw.cfg --site-keyfile  ./site.key
./twpol.txt

sudo tripwire --init --cfgfile /etc/tripwire/tw.cfg --polfile
/etc/tripwire/tw.pol --site-keyfile /etc/tripwire/site.key --local-keyfile
/etc/tripwire/ubuntu-local.key
```

This has created the tripwire integrity database, which contains the filesystem snapshot or baseline. This will then be used as a reference point for all file integrity verifications.

L4.4 Run A Tripwire Filesystem Check

Go to the **/etc/passwd** file and change the owner to **fred** with a command such as:

```
/etc$  sudo chown fred:fred passwd
```

Next go to the **/tmp** folder and change the ownership of one of the files.
Next run an filesystem integrity check with Tripwire, sending it to a file and edit the file, with:

```
sudo tripwire --check > check.txt ; vi check.txt
```

☞	What do you observe from the results?
☞	Why does Tripwire not report changes to the file in the temp directory? (check the twpol.txt)

The vi editor can be exited with the **:q** command. (**:q!** if you get stuck)

L4.5 Create A New Tripwire Rule

Go to your home directory and create a new folder, and a file within, using commands such as:

```
cd ~ ; mkdir pizza ; cd pizza ; touch bbq.pizza
```

Run another filesystem integrity check with Tripwire, with:

```
sudo tripwire --check > check.txt ; vi check.txt
```

- ☞ Does Tripwire report the creation of the new directory?
- ☞ Why?

Go back to the tripwire directory, and add a rule to the policy file for Tripwire (**twpol.txt**), to monitor the pizza directory, using SEC_CRIT. See Unit 2 in the Handbook for guidance.

- ☞ Tripwire Rule:

Next run the following commands to recreate the tripwire policy file and the tripwire database (use **napier123** as the site and local encryption keys):

```
sudo twadmin --create-polfile --cfgfile ./tw.cfg --site-keyfile  ./site.key
./twpol.txt

sudo tripwire --init --cfgfile /etc/tripwire/tw.cfg --polfile
/etc/tripwire/tw.pol --site-keyfile /etc/tripwire/site.key --local-keyfile
/etc/tripwire/ubuntu-local.key
```

Run another filesystem integrity check with Tripwire, with:

```
sudo tripwire --check > check.txt ; vi check.txt
```

- ☞ Does Tripwire report the creation of the new directory?
- ☞ Why?

4.4 Toolkit 4 (Packet Capture/Analysis)

On-line video demo of the toolkit development part of the lab:
http://buchananweb.co.uk/adv_security_and_network_forensics/toolkit04/toolkit04.htm

The objective of this series of labs is to build an integrated toolkit. Open up:

http://buchananweb.co.uk/toolkit.zip

and extract to a local folder. Next open up toolkit.sln, and double click on client.cs. (Refer to http://buchananweb.co.uk/dotnetclientserver.zip for a completed version).

L4.6 Select the [Packet Capture] and then [Open TCPDump] tab, and double click on the "Open TCP Dump" button and add the following code:

```
this.dgPackets.Rows.Clear();
                    PcapDevice device = null;
            Packet packet = null;
            openFileDialog1.InitialDirectory = homeFolder+"\\log";
            openFileDialog1.Filter = "pcap files (*.pcap)|*.pcap|All files (*.*)|*.*";
            openFileDialog1.FilterIndex = 1;
            openFileDialog1.FileName = "*.pcap";
            openFileDialog1.ShowDialog();

            try
            {
                fileName = openFileDialog1.FileName;
                linkLabel7.Visible = true;
                tbFile.Text = fileName;
                device = SharpPcap.GetPcapOfflineDevice(fileName);
                device.PcapOpen();

            }
            catch (Exception e1)
            {
                MessageBox.Show("Error: " + e1.Message);
                return;
            }
            int count = 0;
            while ((packet = device.PcapGetNextPacket()) != null)
            {
                if (packet is TCPPacket)
                {
                    count++;
                    saveTCPPackets(false, count,dgPackets,packet);
                }

                else if (packet is ICMPPacket)
                {
                    count++;
                    saveICMPPackets(false, count, dgPackets, packet);
                }
                else if (packet is UDPPacket)
                {
                    count++;
                    saveUDPPackets(false, count, dgPackets, packet);
                }
                else if (packet is ARPPacket)
                {
                    count++;
                    saveARPPackets(false, count, dgPackets, packet);
                }
                else
                {
                    this.dgPackets.Rows.Add(count.ToString(), "N/K", "", "", "", "", "", "",
"");
                }
            }
            try
```

```
            {
                dgPackets.CurrentCell = this.dgPackets[0, 0];
            }
            catch { }
```

Then add the following code to `saveTCPPackets()`:

```
  try
            {
            string time = packet.PcapHeader.Date.ToShortTimeString();
            TCPPacket tcp = (TCPPacket)packet;
            string srcIp = tcp.SourceAddress;
            string dstIp = tcp.DestinationAddress;
            int srcPort = tcp.SourcePort;
            int dstPort = tcp.DestinationPort;

            ASCIIEncoding utf = new System.Text.ASCIIEncoding();
            string s = utf.GetString(getridofnonprint(tcp.Data));

            CreateMessageForStatusAppend(dg, count.ToString(),"TCP", showflag(tcp),
time.ToString(), srcIp, srcPort.ToString(), dstIp, dstPort.ToString(), s);

            if (realtime == false)
            {
                dg.RowsDefaultCellStyle.BackColor = Color.LightPink;
                for (int i = 0; i < 9; i++)
                {
                    dg.CurrentCell = dg[i, count - 1];

                    dg.CurrentCell.ToolTipText = string.Format("{0}:\r\n{1}",
dg.Columns[i].HeaderText, dg[i, count - 1].Value);
                }
            }
            }
            catch (Exception ex)
            { MessageBox.Show("Exception (TCP) Save:" + ex.Message); }
```

And for `saveUDPPackets` ():

```
            try
            {
            string time = packet.PcapHeader.Date.ToShortTimeString();
            int len = packet.PcapHeader.PacketLength;

            UDPPacket udp = (UDPPacket)packet;
            string srcIp = udp.SourceAddress;
            string dstIp = udp.DestinationAddress;
            int srcPort = udp.SourcePort;
            int dstPort = udp.DestinationPort;
            int packetLength = udp.Length;
            ASCIIEncoding utf = new System.Text.ASCIIEncoding();
            string s = utf.GetString(getridofnonprint(packet.Data));
            //count++;
            CreateMessageForStatusAppend(dg, count.ToString(), "UDP", "", time.ToString(),
srcIp, srcPort.ToString(), dstIp, dstPort.ToString(), s);

            if (realtime == false)
            {
                dgPackets.RowsDefaultCellStyle.BackColor = Color.LightYellow;

                for (int i = 0; i < 9; i++)
                {
                    dg.CurrentCell = dg[i, count - 1];
                    dg.CurrentCell.Style.BackColor = Color.PowderBlue;
                    dg.CurrentCell.ToolTipText = string.Format("{0}:\r\n{1}",
dg.Columns[i].HeaderText, dg[i, count - 1].Value);
                }
            }
            }
            catch (Exception ex)
            {
               // MessageBox.Show("Exception (UDP): " + ex.Message);
```

```
        }
```

L4.7 Now do the same for `saveICMPPackets()` and `saveARPPackets()`. Using a similar code.

L4.8 Select the [Packet Capture] and then [Packet Capture] tab, and double click on the "Start Capture" button and add the following code:

```
            packetcount = 0;
            dgPackets1.Rows.Clear();
            enableNewInterface();
```

and add the following code to enableNewInterface():

```
            packetcount = 0;
            try
            {
                if (comboBox2.Text == "" || comboBox2.Text.StartsWith("-") )
                {
                    CreateMessageForStatus(tbPacket, "Interface not set yet!");
                    return;
                }
                if (device != null)
                {
                    device.PcapStopCapture();
                    device.PcapClose();
                    CreateMessageForStatus(tbPacket, "Interface disconnected");
                    device = null;
                }

                    PcapDeviceList getNetConnections = SharpPcap.GetAllDevices();


                    NetworkDevice netConn =
(NetworkDevice)getNetConnections[comboBox2.SelectedIndex];

                    device = netConn;
                    CreateMessageForStatus(tbPacket, "Network connection: " +
device.PcapDescription+"\r\n");
                    device.PcapOpen(true, 500);

                    device.PcapOnPacketArrival +=
                          new SharpPcap.PacketArrivalEvent(device_PcapOnPacketArrival);

                    device.PcapSetFilter(tbFilter.Text);
                    CreateMessageForStatusAppend(tbPacket, "Filter: " + tbFilter.Text +
"\r\n");
                    device.PcapStartCapture();
            }
            catch (Exception ex)
            {
                CreateMessageForStatusAppend(tbPacket, "Problem opening connection. Error: " +
ex.Message);
            }
```

L4.9 Test the program for opening TCP dumps, and for packet capture.